Cybernetics Technology Office Demonstration and Development Facility (CTO/DDF): Final Technical Report

AD A107663

LEVEL II

Technical Report
CCA-80-15
December, 1980

Lynn Brock

Computer Corporation of America
575 Technology Square
Cambridge, Massachusetts 02139

Cybernetics Technology Office Demonstration and Development Facility (CTO/DDF): Final Technical Report

Lynn Brock

Technical Report
CCA-80-15
December, 1980

This research was sponsored by the Defense Advanced Research Projects Agency under ARPA Order No. 3664 and Amendment Number 1; Contract No. MDA903-79-C-0012. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

Table of Contents

1. Introduction

Computer Corporation of America (CCA) has completed a one-year project designed to continue and expand the operation of a Demonstration and Development Facility (DDF). The DDF was a first-of-a-kind research facility used to integrate, demonstrate, and transfer computer-based research products. The DDF also provided computer resources and support services for the development of entirely new research products.

The DDF began in 1977 under funding from what was then the ARPA Cybernetics Technology Office (CTO). It commenced full operation ahead of schedule and completed a highly successful initial program of product demonstration, transfer, integration, and development support. During its first year, the DDF devoted itself exclusively to CTO-funded Crisis Management research.

Beginning with fiscal 1979, CCA expanded the scale of the DDF operation to serve researchers working on all CTO programs. CCA also conducted intensive technology transfer efforts in support of four selected CTO research programs: Combat Readiness/Effectiveness, Advanced Decision Technology, Command Systems Cybernetics, and Crisis Management. The intensive transfer support for the Crisis Management Program represented a continuation, on an expanded scale, of DDF activities which had already met with success. In the three other selected programs, the activity was new to the DDF but closely related to previous CCA work.

The DDF had the basic purpose of increasing the return on CTO expenditures by accelerating the creation of high-quality, concrete, transferred research results. This was achieved through:

A. Sharing hardware and software resources.

B. Providing expert support and consultation on computer-related matters.

C. Providing a central facility to maintain, integrate, demonstrate, and store finished computer-related products.

D. Exercising the capability for the technology transfer and support of research results in experimental use.

Some specific benefits to the CTO were:

- Minimizing the development of redundant and/or incompatible hardware/software products.
- Demonstrating multiple research products on a single visit (residing on a single menu-driven system), even when those products were developed in diverse, independent environments.
- Integrating research products into consolidated packages for well-coordinated and coherent demonstrations and transfer.
- Convenient access for researchers to each other's products and data, by virtue of their initial development at, or eventual integration into, the DDF.

1.1 Problems of Computer-Based Research Products.

A frustrating problem exists in the field of Cybernetics Technology, in that research results often become at least partially embodied in computer software or computerized databases. The usual difficulties of validating research results through experimental use are exacerbated. The reason is that computer-based products are diverse, complex, expensive to maintain, and difficult to use in their preliminary forms. Worse yet, the appropriate personnel in DoD who could benefit from them often never see the demonstrable products developed at individual research

centers, simply because they lack the time or travel funds to visit the many locations at which contractors do their work.

While research results are widely publicized in journals, the managers and commanders who must use new technologies often respond much more readily to effective demonstrations and personal contact. This is especially important when the underlying concepts are advanced and difficult to appreciate except when seen in action.

An initial glance at the problem raises the temptation to solve it merely by asking researchers to deliver more finished products. Then potential users could easily try them and the additional development required to put them into routine use would be reduced. Unfortunately, this approach is prohibitively expensive. In commercial development of computer software, one well-known rule of thumb says that a finished program product costs roughly nine times as much to develop as a working prototype [BROOKS]. When funding is fixed, the approach of demanding more finished products would, in effect, reduce significantly the number of programs which get to the experimental stage. Even if it worked, such an approach would not overcome some of the other technical obstacles affecting research product demonstration and transfer.

Research products are often:

- Implemented in a variety of languages.
- Implemented on a variety of hardware configurations.
- Saved on a variety of media, in varying formats, at dispersed locations.
- Minimally documented.
- Designed without full awareness of, or concern for, software engineering practices.
- Not fully or systematically tested.

- Developed with little or no regard for the operational environments to which they might later be transferred.

The diversity of forms among, and the common defects of, research products naturally results when each researcher employs the most direct approach to a specific goal. People use available skills, software and machines to get quick, initial results. Indeed, it is precisely such economies that help make the initial proposals attractive to funding agencies.

However, problems arise when the best of such results are to be transferred for experimental use. At that point, one faces the initial technical barriers. For example, suppose that the target military system is a PDP-11 computer on which only COBOL and FORTRAN are maintained, while the research was done in APL on another computer (incompatible hardware and languages). This is a second kind of problem - same hardware/software, but new features. A few new features are needed to install even an experimental system, but the original implementor has begun working on the development of a new concept. Furthermore, he has not documented the code (because he wanted to keep the cost of the initial development down) nor even fully tested and debugged the software.

In addition, a few organizational problems arise:

- Support of an experimental installation involves different skills and motivations than development of an initial prototype.
- Maintenance and distribution of documentation, software and data involve a certain amount of overhead, not justified in many small research

organizations and not economical for a single project.
- The form in which a product can be convincingly demonstrated to a fellow researcher is usually not the best demonstration format for a pragmatic and skeptical user who is expert in a different field.

All of these problems could have been overcome for any given project with sufficient attention, imagination, management and resources. Indeed, funding agencies repeatedly have to do exactly that. However, it seemed valuable to find an alternative solution which would enhance the quality of research results, the number of successful transfers, and the overall cost-effectiveness of research expenditures.

1.2 The DDF Solution

A single concept, embodied in the DDF, simplified many of the above problems in research product development, integration, demonstration, and transfer. That concept was to operate a centralized facility at which the following resources were available:

- Computer hardware and software to support development, testing, and integration of research products.
- Relevant research and software and data produced in previous projects.
- Suitable devices and physical facilities for demonstrations (e.g., advanced graphic displays, audio equipment, specialized terminals).
- Expert hardware and software support staff, familiar with the research areas and aware of existing products.

The most valuable of these resources, the support staff, aided in the following critical functions:

- Installing and integrating research products into the DDF.
- Developing new research products at the DDF.
- Developing effective demonstrations.
- Conducting demonstrations for visitors.
- Transferring research products out of the DDF into the field.
- Supporting experimental use of research products.
- Modifying existing products to enhance their transferability.

The use of a centralized facility and staff dedicated to these functions provided the following advantages:

- A single source for processing, software, data and consultation in the development of new computer-based research products.
- A single location for the demonstration of multiple products.
- A single source, supporting uniform hardware and software, from which to stage transfers.
- An organization familiar with the mechanics of transfer and the products to be transferred for the support and planning of transfers.
- Elimination of one integration step in some future developments, since these were planned from the outset for implementation at the DDF.
- Uniform procedures for the documentation, distribution, maintenance and support of products in experimental use.
- A single source of information for ARPA on the progress, status and problems in various transfer experiments.
- A well organized library of research products and documentation which itself assumed increasing value as use of the DDF proceeded.

As a general assessment, the DDF provided the means to develop and transfer the results of computer-based projects more reliably, more economically, more frequently and more successfully than any other available approaches. Though a relatively new concept at its beginning, the DDF was tested during 1977 on one key program - Crisis Management - before expanding to other programs. CCA's experience with the DDF, and its results, form the subjects of Sections 2-4.

1.3 A Dual DDF

The DDF not only reached a fully operational state earlier than originally planned, but it also experienced a heavier level of usage and supported a fuller demonstration schedule. The usage of the system for development and tests associated with technology transfer was very heavy. There arose a tendency for this kind of usage to disrupt, and to some degree conflict with, the demonstration schedule.

To solve the problem, it became necessary to expand the DDF by adding a second computer system, integrating it into the existing configuration so as to maximize resource sharing. That permitted DDF time-sharing users to access the system at any time without having to log off for priority demonstrations or for other scheduled conflicts. A dual facility also expanded basic research capabilities and permitted sophisticated product integration and transfer activities. The nature of the requirements for this dual DDF and its outcome are treated in Section 5.

2. The ARPA/CTO Program

Having surveyed the general DDF concept, this report discussed a second element of background, a brief synopsis of the overall CTO program.

As of early FY79, CTO was sponsoring research efforts in the following areas:

Advanced Decision Technology

The Advanced Decision Technology Program focused on developing computer-based models for accessing, evaluating, formulating, and implementing Defense decision processes as well as solving the difficult problems of allocating Defense resources and real-time decision-making efforts. Progress in this program was shown most clearly by extending the capability of national security planners and analysts to perform essential forecasting, intelligence, and decision functions crucial to deterrent posture, readiness starts, force composition and development, and crisis management. It also enhanced the DoD understanding of adversary intentions and capabilities. Decision analysis techniques were used in more than thirty military applications.

Command Systems Cybernetics

The Command Systems Cybernetics Program addressed the problem of increased man-computer system productivity and cost-effectiveness in a command and control context. The program developed tools for spatial information management, reasoning assistance, and communications augmentation. Specific projects involved the demonstration of spatial information management capabilities, including spatial information storage and retrieval, adaptive information filtering, information pacing, structuring, ordering, reasoning assistance tools for plausible reasoning, heuristic modeling, and decision aiding. Finally, the Command Systems

Cybernetics Program developed communications augmentation tools for group decision-making and group problem-solving.

Crisis Management

The Crisis Management Program had the primary goal of developing computer-based systems for monitoring, forecasting, and managing international and national affairs, especially crises. Progress included the development of a real-time monitoring and warning system, composed of quantitative political indicators, a short range forecasting capability, and an interactive computer database for the investigation of crisis parameters. Progress also included the development of three crisis management executive aids, including an aid to assist commanders in option selection and implementation, the identification of crisis management problems before and during the occurrence of important international events and crises, and the development of an interactive database system which enables commanders to search rapidly for crisis management precedents. Lastly, progress was made in the areas of quantitative indicator development, quantitative forecasting, methodological development, and man-machine interfacing.

Cybernetics of Instructional Systems

Research in this area had the goal of developing and evaluating computer-managed instructional systems in an attempt to significantly decrease the cost of Defense training activities. This area of research focused explicitly upon the development of advanced instructional systems to enable DoD to meet its crucial, large-scale training requirements more efficiently and at the highest possible quality standards.

Logistics Systems Technology

The Logistics Systems Technology Program aimed at reducing costs of logistical procedures and performance in the areas of commercial commodity acquisition, maintenance incentives, and procurement. More specifically, the Logistics Systems Technology Program attempted to improve logistical performance through synchronizing maintenance incentives with organizational structures, and to improve procedures through the evaluation of commercially acquired material and competitive procurements. As with nearly all CTO programs, the results of this research were incorporated into interactive computer programs designed to provide on-line, real-time decision-making support to DoD personnel.

3. CTO/DDF Core Activities

Operation of the CTO/DDF involved a technical program with activities in three basic areas:

- Development
- Demonstration
- Support

Development centered on making existing products of CTO-funded research more immediately transferable or more widely usable on an experimental basis.

Demonstrations were conducted so that members of the DoD community could assess the research results and products with respect to their operational needs.

Support to ongoing research programs allowed their software to be developed in a framework which ensured the integration of future products into the DDF, and prepared for their eventual smooth transfer to operational DoD units.

This section presents the technical activities and projects of the CTO/DDF. In general, these consisted of the following core activities:

A. Basic services, which included such tasks as operating and maintaining the hardware and systems software for the facility.

B. CTO Program support activities, which included such tasks as product demonstration, transfer, and documentation assistance. These tasks were distinguished from basic services in that they involved knowledge by CTO/DDF personnel of specific CTO efforts. This specific knowledge represented a unique resource associated with the DDF which

increased in value as development and technology transfer proceeded.

3.1 Basic Services

CCA managed and operated the site, hardware, and software facilities of the CTO/DDF. The CTO/DDF used the physical site previously set up for the CTO/DDF and incorporated the hardware and software configurations of the earlier experimental facility. Thus, many of the preparatory requirements for the CTO/DDF had already been satisfied.

In order to fulfill the demands of the CTO/DDF, it became necessary to plan for growth in terms of staff and equipment. It was important for the facility to have available sufficient hardware and software resources to ensure the readiness of the CTO/DDF to support the research efforts of all CTO contractors. This involved the procurement of equipment with the advice and approval of CTO management. It also included the establishment of new operating procedures and personnel to administer those procedures.

Basic services tasks covered the management of three CTO/DDF resources: the hardware configuration, the system software, and the facility's library.

3.1.1 Hardware Configuration

The CTO/DDF used the hardware configuration procured for the Crisis Management/DDF, and expanded upon it as needed. The final configuration is summarized below.

The DDF received a Digital Equipment Corporation PDP-11/70 computer system, associated software, terminals and modems as government-furnished equipment (GFE). The DEC PDP-11/70 had 512K (8 bit) bytes of memory. Other options included:

- FP 11-C floating point processor.
- Kw 11-L line frequency clock
- M9301-YC bootstrap loader

The peripherals included:

- TJU16 tape transport (1)
- RK05 disk drives (2)
- RP04 disk drives (2)
- LV-11 electrostatic printer/plotter
- LA-36 Decwriter II consoler typewriter
- DH-11 multiplexors (2)
- Astroset Modem Cards (6) 300 bps, (6) 1200 bps
- Tektronix 4051 interactive graphic systems (4)
- Tektronix 4025 terminal
- Anderson Jacobson AJ832 terminal
- UNIVAC 1652 terminal
- Texas Instruments TI-745 portable terminals (2)
- Tektronix 4631 hard copy units (2)

The DDF also had an IBM 5100 computer in use for the design and implementation of the Marine Corps Combat Readiness Evaluation System Software Application (MCCRESSA), and for the demonstration and transfer of other decision analysis software.

In support of the communications requirements, the CTO/DDF provided twelve local dial-up lines in the Washington area. There were two rotored hunt-groups of six lines each, one at 300 baud communications and the other at 1200 baud. In addition, the facility had a "DDD" line to California. This line allowed access to the computer from either the University of Southern California in Los Angeles or the Naval Postgraduate School in Monterey.

Obviously, the CTO/DDF was responsible for a large amount of sophisticated and up-to-date equipment. The terminals and modems had been made available at the facility as well as distributed (at CMP management direction) to user sites to enhance their in-house capabilities. However, the responsibility for the equipment remained with the CTO/DDF as the central distribution point.

In addition to the configuration presented above, sufficient new equipment was procured by CCA as required by CTO to support and enhance program operations during FY79:

- IBM 5110 computer as an upgrade of the IBM 5100 in the present configuration
- RP04 disk to reduce contention and accelerate swapping, thus increasing system capacity
- Tektronix 4027 color graphics terminal

The hardware configuration of the CTO/DDF included an IBM 5100 computer used for the Combat Readiness/Effectiveness programs and other decision analysis demonstrations and transfers at the facility. These efforts involved the design and implementation of the Marine Corps Combat Readiness Evaluation System Software Applciation (MCCRESSA). The users of this technology, the Marine Corps, obtained several IBM 5100 computers to support MCCRESSA in FY79. In order to continue its support of the Combat Readiness/Effectiveness program, CCA leased a number of IBM 5110s. This was necessary for continued development efforts and also for demonstration and transfer activities involving MCCRESSA.

The equipment procured to enhance the system was subjected to stringent tests to ensure that it fulfilled the needs of the CTO program, for it was essential that CCA produce the optimal configuration for the CTO/DDF. As the CTO program evolved and greater demands were placed on the CTO/DDF,

additional equipment needs developed. CCA investigated procurement alternatives to satisfy these requirements, and initiated appropriate action as directed by CTO management.

3.1.2 System Software

The software system for the CTO/DDF consisted of the UNIX operating system and a number of additional software products. As a result of CCA procurement efforts, the configuration included:

A. UNIX multi-programming time-sharing operating system
B. CULC FORTRAN IV Plus
C. Tektronix PLOT 10 and PLOT 50 Software
D. INGRES, a database system
E. Rand editor NED
F. RITA--Rule-directed Interactive Transaction Agent

In addition to these packages, CCA personnel developed several pieces of specialized software utilities to support R&D needs. These included:

- Tektronix 4051/PDP-11 parallel processing capabilities
- Half duplex terminal drivers

3.1.3 Operation of the Time-sharing Device

Providing time-sharing computer services to CTO contractors was the most basic function of the CTO/DDF. Several of the activities comprised this function:

A. Operating the system according to schedule -- the time-sharing service remained in operation 24 hours a day, seven days a week. One shift of eight hours was attended operation with the remaining time being unattended operation. During periods of intense

CTO/DDF usage, CCA increased the amount of attended operation so far as was possible within the facility's level of staffing.

B. Isolating and correcting hardware and software errors -- Correction of hardware errors was accomplished primarily through maintenance agreements with suppliers. Maintenance agreements were purchased from vendors (e.g. DEC and Tektronix) to cover the computer mainframe and graphics units. Hardware failures were normally investigated first by CCA staff; those that could not be diagnosed and repaired quickly were referred to appropriate maintenance suppliers for service.

Aside from routine hardware failures inevitable in the operation of any computer system, two problems and their solutions deserve special mention.

- Recurrent disk errors, whose cause remained unclear despite the vendor's repeated attempts to locate the problem, were effectively dealt with by modifying the disk support software within the operating system to print out more complete and readable diagnostic messages. Eventually, the vendor did locate the difficulty, and repaired it.

- Sporadic memory failures, possibly caused by a memory not maintained by the primary vendor, caused a good deal of downtime. DDF personnel responded by coordinating the efforts of the vendors involved and by instituting procedures which allowed a large portion of the DDF user community to accomplish their projects on the partially disabled machine.

Software errors were resolved in various ways deperding on the category of software involved. Because vendors supported much of the CTO/DDF's system software, they corrected any errors reported in it. Software change notices from vendors (i.e. reports from vendors describing error corrections) were applied promptly by CCA personnel to the facility's software. However, CCA or sources other than established vendors, developed some of the CTO/DDF's system software, and errors in this category of software were corrected by CCA.

Applications software, on the other hand (that is, software developed by CTO contractors) fell into two categories. The first was software maintained by the developer; most applications packages used at the early stages of the CTO/DDF belonged to this category. The second was software of general utility turned over to CCA for maintenance; as time went on, this category of software became increasingly more common. However, when software errors proved especially disruptive to facility operations, CCA personnel endeavored to diagnose and correct them regardless of software support category.

Two instances of software tailoring should be highlighted:

- The BMD-P series of programs was installed at the facility and made generally available. Various individual BMD programs were modified at the request of several DDF users.

- PWB/UNIX software components were retrofitted to the existing operating system, to lay the groundwork for an orderly transition to PWB/UNIX at a later date.

C. Performing preventive maintenance on hardware -- A program of preventive maintenance was undertaken in accordance with manufacturers' specifications. This program embraced the major computer hardware components and the facility's computer room air conditioner. Preventive maintenance was performed approximately once per month.

D. Backing up stored data and programs on a daily and monthly basis -- A policy of regular and frequent backup of stored data was followed in order to protect stored data and programs against catastrophic failures of hardware or software.

3.1.3.1 User Support

The CTO/DDF, through its hardware and software configuration, offered extensive computing capabilities to CTO researchers. The purpose of user support activities was to ensure that these extensive capabilities were used to their utmost, and that each capability delivered its maximum benefit to the CTO program.

The central functions of user support were:

- To ensure that all users knew what facilities were available on the system and how to use them.
- To advice users in the selection and utilization of CTO/DDF capabilities.
- To help users overcome difficult and obscure bugs caused by errors in system hardware or software.

Through user support, the CTO/DDF helped ensure that contractors took full advantage of the latest technology available and helped CTO management in directing its contractors to use new technology. These benefits were

achieved through a series of CCA activities that included:

A. Documentation of system capabilities -- CCA provided documentation to CTO/DDF users describing the capabilities of the CTO/DDF computer system and how to use those capabilities. Also, CCA made available to CTO/DDF users the documentation distributed by hardware and software suppliers regarding their products. This included user manuals for languages at the facility and manuals on user interface equipment (graphics units, etc.). Altogether, roughly 50 commands or programs were added to the system; the following example illustrates the formal and organization of a typical document.

B. Seminars on system capabilities -- CCA held informal seminars for CTO/DDF users to familiarize them with the system and to respond to questions. One important motivation for holding seminars of this type was to introduce users to capabilities of the CTO/DDF that had not been previously available to them. These seminars were held at the facility in its demonstration area, and also served to acquaint CTO contractors with the physical site itself. In addition to numerous informal seminars held for particular contractors, a more formal seminar describing general system capabilities was given.

C. Advice on CTO/DDF Capabilities -- CCA advised CTO/DDF users on the selection and utilization of CTO/DDF capabilities, as well as on general problems with program design and construction. This typically involved matters such as selecting appropriate languages and system utilities to use, choosing efficient database access techniques and the like. Several CTO contractors received design suggestions

based on specific DDF capabilities, which resulted in reduced implementation time, and increased effectiveness for their efforts.

D. Debugging assistance -- At the request of users, CCA provided assistance in diagnosing errors in their programs. The purpose of this activity was to place computer experts at the disposal of CTO contractors to help them in correcting exceptionally difficult errors. This activity also provided a vehicle for identifying obscure errors in system hardware and software, and for assisting users in circumventing those errors until they were fixed. Assistance in this vital area was given on an almost daily basis to virtually every CTO contractor. Major debugging efforts occurred in the installation of the PRESS program and the BMD-P series of statistical programs.

3.1.4 Tailoring

Another aspect of the total computation support provided by CCA through the CTO/DDF was the tailoring of CTO/DDF capabilities to match the precise needs of CTO researchers. It often happened that the R&D staff required and/or benefited from the development of specialized utility or system software. For example, a researcher developing highly interactive graphics software might require more responsive I/O drivers than are normally supplied. Or, an application that made heavy, repeated use of a database might benefit from specialized indexed access methods.

3.1.5 Facility Library

A library to house various kinds of reference material was built for the CTO/DDF. It contained copies of user manuals, technical reports, sample output, reference manuals for system software, and other documentation for all CTO products

under development and supported by the facility. In addition, the library contained a large collection of digital magnetic tapes, which was managed with the aid of a specialized database.

3.1.6 Programs Support Tasks

In addition to operating and maintaining the CTO/DDF, CCA provided special program support for CTO research software. These support tasks included packaging and demonstration of product software.

3.1.7 Packaging of Software and Data

As part of the packaging support activity, CCA assisted contractors in the final testing and documentation of useful software and database products developed at the facility. CCA then notified facility users of the existence of such products, and provided assistance for using them through documentation, seminars, and consultation. Also, CCA maintained packaged products in cooperation with the contractors who developed them.

3.1.8 Demonstrations

In its role as a centralized demonstration facility, the DDF provided a supportive environment for presenting CTO research efforts in an effective manner. The packaging of software and data at a centralized facility allowed the products of different contractors to be integrated for a coherent demonstration. Demonstrations were conducted either by CTO contractors or management, or by CCA personnel under the direction of CTO management. Before demonstrating a product, CCA personnel received training from the developer on the use of the product; then they assisted in each demonstration to the extent required by the CTO.

3.1.9 Intensive Technology Transfer Programs

Certain programs received more intensive support from DDF under the CTO/DDF program. In each case, a project was formulated to facilitate transfer of some developed CTO technology. These projects involved software development or conversion, documentation, and related activities. In all cases they started from an existing research product and made the changes necessary to move the product out of the research environment into the DDF and the field. The programs receiving intensive support for technology transfer were:

- Advanced Decision Technology
- Crisis Management
- Command System Cybernetics
- Combat Readiness/Effectiveness

Over the past several years ARPA has sponsored the development of various automated decision aids utilizing advanced decision technology. Prototype versions of these computer programs were used successfully in treaty negotiation, as evaluation tools to aid in the procurement of large systems, as posturing aids, and in other applications. At the start of the CTO/DDF program, these programs existed in experimental form in APL for the IBM 5100 computer line. CCA has since extracted the techniques and algorithms used in the prototype models and produced new versions of a selected set of these programs, written in FORTRAN IV for the PDP-11/70 computer.

The next section presents a brief description of the set of programs which currently makes up the ADT software.

4. ADT Software Program Descriptions

A. EVAL and DISC-EVAL

This software allows users to construct hierarchical decomposition evaluation models for the evaluation of complex systems. The user interactively provides the structure and labels, and assigns importance to them. The software supports simultaneous comparison of up to five systems. Output of the software is the unit of merit (score) for each candidate being evaluated. Besides the final score, the software can display intermediate aggregation as well as discrimination at each level. It also produces a "roadmap" which shows the key discriminators which most significantly differentiate the contending systems.

A sensitivity analysis allows the user to determine the criticality of sets of important factors.

A database retrieval capability can be used to store descriptive summaries, making EVAL a useful briefing tool for high level decision makers.

Prototype versions of this software were used successfully in procurement cycles of the improved TOW Vehicle, shipboard intermediate range combat system, the single-channel ground-to-air combat systems for the Department of Defense, and for other system evaluations such as evaluation of the United States Military Academy.

B. DECISION

This software allows users to construct, interactively, decision trees using four basic types of combinatorial rules: probability nodes, simple cumulative nodes, multiplicative nodes and decision nodes. The primary objective of DECISION is to model a decision, or some part of it, so that at least some of the implications can be deduced.

C. OPINT and IMPROVED OPINT

This software provides computer-driven option screening and intelligence assessment. Using multi-variate decision techniques, the software generates an expected value matrix of option selection. This technique aids decision making in situations where the key state variables are not known.

OPINT provides dyadic (two-factor) influence diagramming capability to aid decision makers to select from various related and uncertain options. The program includes tutorial information so that it can be used by casual users.

The prototype version of this software aided decision makers in selecting the best posturing option for the Sixth Fleet during the Lebanese evacuation crisis. It has also been used during various planning exercises throughout the European Command (EUCOM/J2, J3).

D. INFLUD

This program extracts the dyadic influence diagramming capability of OPINT and makes it available separately for use by the intelligence community. This capability allows the analyst to decompose an intelligence problem into separate components which are easier to analyze than the overall likelihood of the event in question. After the analyst makes the individual assessments, the computer program reconstructs the problem using conditioned probabilities and assesses the overall likelihood that an event will occur.

E. INFER

INFER is an inference modeling system which aids the user in building probability diagrams of hierarchical inference. These are most useful when the complexity of a real-world inference problem requires an amount or kind of knowledge beyond the capability of any one individual. In such cases,

many different individuals with different expertise can decompose the problem along hierarchical lines, assessing those probabilities which link the data through intermediate variables to the main hypothesis.

E. POM

POM, or Program Objectives Memorandum, assists in the budgeting cycle. Previous budgetary efforts have been oriented toward procurement of the items, considering potential effectiveness of the items; POM generates a profile of items under budgetary consideration, considering both cost and effectiveness.

In addition to the profile, this software allows the user to capture the rationale used in determining effectiveness measures. POM can be used to easily perform trade-off analyses by grouping packages of funding efforts so that they can be compared on the basis of cost-effectiveness.

G. PARETO - OPTIMAL TREATY ANALYSIS

This system provides a mathematical way of objectively generating trade-offs for treaty negotiation. The software implements the Pareto optimal curve, which takes dozens of issues into consideration and determines optimal treaty conditions. The prototype version of this software was used in the Philippines Base Rights negotiations.

H. SCORING RULE

This software implements a computer-based scoring rule training procedure which helps to calibrate and improve the accuracy of probability assessors. In application, the computer poses a series of multiple-choice questions to the trainee. The trainee is required to indicate the correct answer along with a probabilistic assessment of his degree of certainty about a designated answer. The program provides

automatic feedback as to the accuracy of the trainee's response and the computer maintains a running calculation of the implications of cumulative response accuracy and uncertainty levels to determine and display the degree of sorting and labelling error in the trainee's performance.

4.1 Crisis Management Program

The CTO Crisis Management Program (CMP) was a major effort to develop, test, and transfer technology in three areas;

A. Computer-based early warning and monitoring systems.
B. Computer-based executive aids for crisis management.
C. New quantitative methods for advanced warning, monitoring, and management.

Wide-ranging research has been directed toward each of these areas by ARPA since 1974. Initial work through 1976 was directed toward certain basic research themes that are prerequisites for effective technology development in the social sciences. Later work was directed at producing user-oriented, computer-based aids to:

- Assist defense operations centers in identifying what indicator and warning patterns signal the onset of a crisis.
- Develop option generation and evaluation aids to assist crisis managers after a crisis has begun.

The Crisis Management Program (CMP) received intensive support with good results following the creation of the DDF in 1977. In fact, the experimental CM/DDF served to test the concept of a DDF. CCA then expanded its support of CMP research efforts through the CTO/DDF. The support provided by CCA included four tasks:

- Continued support of intelligent Indicators and Warnings (I&W) capability.
- Support and coordination of the integration of the Group Decision Aid and Executive Aid programs.
- Support in the development of a system to aid in the management and support of the transfer of crisis management technology to the DIA/NMIC, NPS, and other sites.
- Provide services for a variety of CMP projects in the areas of user support and data services.

One of the most successful crisis management tools developed was the Early Warning and Monitoring System (EWAMS) [ANDRIOLE], [WITTMEYER]. This program computed and displayed quantitative indicators of international activity and tension based on summary information extracted from news stories. Tests of EWAMS demonstrated that it can provide important indicators of impending crises. The program might have significantly improved the defense community's ability to forecast earlier crises, such as the 1967 Sino-Soviet border clash, the 1968 Czechoslovakian invasion, and the 1971 Indo-Pakistani war.

Operation of the EWAMS program allows an analyst to inquire about the interactions between two nations or two groups of nations. However, the program as it stood did not automatically recognize that a given nation pair was exhibiting unusual tension or conflict activity and suggest that the analyst examine their interactions more closely. During EWAMS demonstrations, key commanders and managers in operational intelligence organizations indicated that this sort of "alerting" service would be enormously useful to their operations.

Accordingly, CCA continued its support of the development of an intelligent I&W program. Specifically, CCA staff:

A. Assisted the CMP contractor who developed the intelligent I&W capability in design and coding of the software. In particular, such participation is useful for interfacing the program with the UNIX operating system.

B. Provided aid and recommendations during the quality assurance and testing phases of the program.

C. Provided advice for real-time execution of the capability.

Two important software products of CMP research efforts were the Executive Aid for Crisis Management [CACI] and the Group Decision Aid Programs [LEAL et al].

The Executive Aid Program was developed by C.A.C.I., Incorporated, to assist DoD crisis managers by providing them with ready access to the historical record of post-war U.S. crisis operations. The data file manipulation capabilities of the aid allowed crisis managers to search rapidly for historical precedents and analogies in the course of considering crisis options. In noncrisis periods the aid served as an instructional device for crisis management personnel. It was used to outline the history of U.S. crisis management activity since World War II, to summarize crisis problems that the United States has faced in the past, and to identify recent trends in problems faced by U.S. crisis managers, and thereby plan contingencies.

The database for the Executive Aid contained 307 crises involving the United States during the period 1946-1976. The crisis cases were coded to index their major characteristics; subsets of the set of 307 were subjected to more intense analysis and were coded to produce more detailed databases

for special applications. Three crisis databases were produced for use in the Executive Aid:

A. A file of 101 U.S. crisis operations during the period 1956-1976 which focused on U.S. actions and objectives during the responses.
B. A sample of 41 crises involving the United States during the period 1956-1976, which presented the major crisis management problems that the United States encountered in these operations.
C. A set of 307 U.S. crises over the period 1946-1976, which provided descriptive information concerning U.S. military management during each incident and presented a selected set of general crisis descriptors.

The interactive, self-prompting, Executive Aid Program allowed a user to search the crisis history files and review crisis information. Crises were presented for review based on user-selected attributes.

4.1.1 The Group Decision Aid

The Group Decision Aid was developed by Perceptronics, Incorporated as a system for interactive computer aiding of group decision making. It featured simple individual data entry terminals and a large-screen, color video display for feedback of computer-generated information. Its purpose was to guide the group decision making process by selective elicitation of a decision tree which incorporated value and probability inputs from all group members. A specially trained system operator, called an intermediary, facilitated group interaction with the aid program so that group members needed no prior familiarity with computers or decision analysis.

This tool was successfully tested under a crisis (terrorist) scenario and later transferred to the CTO/DDF. CCA supervised and aided in the process of transferring both software and hardware for the Group Decision Aid to the CTO/DDF. Once it was installed, CCA operated and maintained the system, and provided facilities for demonstration and later transfer of the tool to DoD operational agencies.

The graphics contained in the software occur in the Crisis Analyzer modules for both the U.S. and the U.S.S.R. These graphics consist of bar graphs representing user-specified subsets of crisis data.

The Tektronix Terminal Control System (TCS) produced the graphics display from Fortran, its output being compatible with Tektronix 4010 Graphics. This means that the software can run on a number of different terminals, such as the Tektronix 4025, 4027, 4051, and appropriately upgraded Lear-Siegler ADM3s.

4.1.2 Terrorist Research and Analysis Program (TRAP)

An ongoing project planned for CMP research efforts was the development of a Terrorist Research and Analysis Program. The program used a database containing information on:

- Associated background conditions.
- Profiles of terrorist groups, intentions, and targets.
- Past terrorist incidents.

The program permitted identification of terrorist incidents and searched for precedents in the database.

CCA supported the development of this system through the CTO/DDF. Advice, computer services, and programming aid were provided for CMP contractors involved in this project. CCA

aided in the testing and documentation of the product and integrated the program into the CTO/DDF system.

4.1.3 General Support of Crisis Management Research

A final task for CCA staff in the Crisis Management area was the supervision and provision of services for user support and data collection, input, storage, and retrieval. CCA provided these services to all CMP contractors at the direction of CTO. Specifically, CCA provided:

A. CMP hotline -- An analyst was available by phone at all times for CMP contractors who required aid in their dealings with the CTO/DDF. The analyst provided advice to help CMP researchers diagnose and solve problems in using crisis management products and other services at the facility. If the analyst on call was not directly responsible for the component at fault, the proper CCA staff member was located and advised of the trouble.

B. Support of special software for Crisis Management efforts -- CCA maintained statistical, database management, and other utility software packages used solely for Crisis Management R&D efforts.

   CCA provided documentation and advice on use of these programs and assistance in correcting problems that CMP contractors had with them.

C. Support for packaging, demonstration, and transfer -- The CTO/DDF staff prepared CMP products for demonstration and transfer. They also contributed to the actual demonstration and transfer efforts for these programs. The Crisis management team was responsible for dealing with aspects directly related to the Crisis Management

R&D program, while the CTO/DDF team handled mostly the underlying system work that supported these efforts.

D. Services -- The CTO/DDF assisted CMP contractors in the design of appropriate databases for CMP projects. CCA also provided services for:

- Input of data to the CTO/DDF system
- Real-time and other data collection
- Data storage and retrieval
- Validation of input data

4.2 Command System Cybernetics

4.2.1 Automated Desk System

In support of the Command System Cybernetics program, CCA developed and installed the Automated Desk, an online facility to support a user in organizing, locating and handling computer-resident information and tools. The Automated Desk permits the user to gain the benefits of computerized communication and information handling, while retaining much of the spatial feel and visual orientation of working at the familiar office desk. The Automated Desk system provides a convenient tool for accessing CTO/DDF computer programs and documentation for use in a demonstration context at the DDF. Users can "browse" through a large number of virtual terminal screens, each running instances of various software programs. Related programs and descriptive documentation can be placed "near" each other for convenient location and use.

The Automated Desk is a derivative of the Spatial Data Management System (SDMS), developed at CCA under ARPA funding. SDMS uses advanced concepts in interfacing a user to a database system, using color graphics and multiple displays as aids in searching for, locating, and recognizing

data in a "landscape". The Automated Desk takes the essential SDMS concepts of spatial organization and pictorial identification, and implements them in simple ways on equipment of modest cost. While the objective of SDMS is to explore the near term limits of the technology, providing extremely powerful facilities, the objective of the Automated Desk project is to put some key SDMS-like capabilities into an inexpensive product that runs on widely available equipment, with the goal of operational use in practical DoD applications.

While the Automated Desk has the spatial metaphor in common with SDMS, its actual use is very different; under the Automated Desk, "activating" an object starts a program, or series of programs, that interact with the user through the terminal screen and keyboard. The program(s) can be left running even when another object is activated, thus allowing convenient user interaction with many simultaneously active processes.

The Automated Desk system consists of an intelligent display terminal connected to a host computer. Through the display, the user sees a flat surface like the top of a desk. Objects representing documents, groups of documents and other information sources are arranged on the surface. The user can move his field of vision to browse over areas of the surface, as well as "moving back" to get a less detailed view of a larger area. As on a real desk top, the user employs his naturally developed skills in spatial organization and pictorial identification to store, locate and identify objects. For example, he no longer needs to know exactly what a document is named -- he can remember approximately what it looks like and recognize it when looking in the area where he left it. Unlike a real desk top, the Automated Desk system has the capacity to handle large numbers of documents, as well as other more dynamic information objects, to aid the

user in searching in a variety of ways, and to store the same information object in several places.

The overall effect is a more dynamic environment in which the user has more current information, greater capability for organizing information, and more search power than he would in a typical manual set-up. In these respects, his capability is also far greater than it would be with existing text-oriented systems, because he is coupling his own natural powers of organization and identification with the strengths of a computerized system which integrates many information handling tools.

As part of the Cybernetics Technology Development and Demonstration Facility, the Automated Desk system provides a tool for accessing descriptive information concerning the CTO/DDF, the CTO program, software descriptions, and the software itself. This section of this report presents a detailed example of how the Automated Desk system might be used at the CTO/DDF. The functionality of the system, the objects that populate a user's "information space," and his means of manipulating his environment are presented. The Automated Desk system is a tool of wide application and should enhance advanced research in the area of man-machine interaction in many DoD environments.

4.2.2 Sample Session Using the Automated Desk System

The user sits at an intelligent display terminal. On the screen before him is the flat surface of part of an imaginary desk top, on which objects of various shapes and sizes are arranged. The rest of the imaginary desk surface is off-screen at the moment. A bank of motion control keys, each labelled with an arrow indicating direction, is part of the terminal keyboard. The user presses a key and the picture changes as if the screen were moving over the stationary desk top. New objects come into view from one side and old

objects pass out of view on the opposite side. The impression is one of motion - the same sort of impression created by a passing landscape viewed from a car window. The action can be likened to scanning (with the eye) the objects laid out on a large desk or table top.

Initially, the user is looking the scene over as if from a distance. He can see from one to about fifteen objects at any one time (roughly the number of documents which he can spread out on an ordinary desk top). The images he sees are smaller than the actual objects, but display enough visual cues so as to be recognizable. They have distinctive shapes, sizes, borders, contents and names. When the user comes upon one he would like to examine in more detail, he moves it toward the center of the screen.

He then pushes the "ACTIVATE" key, which begins execution of a program associated with the centered object. Subsequent interaction with the program is through the keyboard, with all the usual amenities of an interactive system being available.

4.2.3 Functional Description

In the previous paragraphs, the sample of user actions and screen images presents some of the more apparent concepts of the Automated Desk System. The following subsections examine the concepts more closely, first by surveying the system's capabilities, and then by discussing the functions of the Automated Desk in detail.

The Automated Desk provides an environment for the performance of online information handling functions. It presents the user with a large flat surface -- an imaginary desk top -- on which he may arrange objects to:

A. Move around over the surface, thereby viewing different portions of it.

B. Activate an object, thereby awakening the process associated with the object -- diverting user I/O from the Automated Desk to the awakened UNIX process.

C. Create objects and place them on the surface where desired.

D. Associate processes with these objects to perform virtually any online information handling function.

E. Delete objects no longer needed.

F. Label objects and later use the label to move automatically and immediately to the object from any point on the surface, without actually traversing the territory between origin and destination.

G. Use maps, landmarks and other navigational concepts as an aid to orientation and organization.

These functions may be used together to solve problems such as the following:

A. Organization, storage and, later, retrieval of documents in a uniquely convenient form -- spread out as if on a desk so that groupings can be seen at a glance and so that the user's visual memory and internalized spatial organization of the information are exploited. Few workers have the opportunity to spread 100 documents out on a table top and leave them there, although doing so will save much time when turning from task to task.

B. Monitoring a large number of concurrently active systems, easily moving from one display to another as required. Applications of this occur in management, dispatching, security, engineering, production control, and computer system development and operations.

C. Convenient, rapid interaction with a large number of people and information systems - perhaps the most vital function in a command environment. Coordinating the actions of a complex organization involves elements of virtually all of the activities listed in (A) and (B) above. In particular, the ability to turn from task to task rapidly - handling interruptions - without "tearing down" and "setting up" one's work space can assume great significance in a tactical situation.

The user of the Automated Desk has five categories of functions available to him.

- Motion across the desk top
- Zooming into the desk top
- Interacting with virtual terminal
- Creating, Deleting and Moving
- Examining the Desk Top Map

The user sends commands to the Automated Desk (AD) through a terminal keyboard which contains all the standard alphanumeric keys plus a Cursor Control Pad and special function keys.

The Cursor Control Pad (Figure 4.1) contains eight directional keys, the Home key, and the Zoom key. The directional keys are marked with arrows pointing to the north, south, east, west, Northeast, Southeast, Northwest and Southwest. These control the direction of travel across the desk top or position the cursor on the screen, depending on

the mode of the terminal. In Scroll mode, one of the functions of the Mode key, the arrow keys cause the data to move across the screen, without changing the position of the cursor. In Terminal Mode, the cursor moves and the data remain fixed. Since there are only two modes, depressing the Mode key changes the terminal from one mode to the other.

The Home key returns the user to the center of the desk top, in the distant view.

All commands other than motion (which use the cursor pad keys) use one of the special function keys (Figure 4.2).

Before the user can activate an object, he must center its virtual terminal on the screen. He can use either the joystick or the directional keys to correctly position the object.

Once the virtual terminal is centered, the user presses the Activate key to initiate its process. To discontinue it, the user presses the De-Activate key.

Cursor Control Pad — Figure 4.1

Zoom	Mode	(Blank)
↖	↑	↗
←	Home	→
↙	↓	↘

Special Function Keys Figure 4.2

Create Object	Create Aid	Center	Update	Delete	Pick	Put	Goto	Map	Activate	De-Activate

---

The first time a person uses the Automated Desk, his desk top is empty. To create an object, the user positions the cursor where he wishes the object's center to be. He then presses the appropriate key:

- Create Object to create an icon and virtual terminal
- Create Aid for a navigational aid

The user cannot continue if there is not enough room. The system clears the screen, then prompts the user for various parameters which define the object. Once the creation is finished, the system returns to the desk surface where it displays the new object.

After an object has been in use for a short while, the user may desire to fine-tune its appearance or mode of operation. When he presses the Update key, the system allows him to change the characteristics of the object that the cursor is superimposed upon. When the key is pressed, the system clears the screen and prompts for new information. To end the update routine, the user presses the Update key again. The system generates the new object and returns the user to his original screen.

As the contents of the desk accumulate, the user may need to rearrange the desk top. He uses the Pick and Put keys to reposition an object. The Pick key sets a pointer at the object marked by the cursor. The Put key transposes the object to the new position without altering it; once the

object is in a legal location on the screen, the system deletes it from the old location.

The user can delete an object by placing the cursor over it and pressing the Delete key.

The user presses the Goto key when he wishes to travel immediately to a labelled object. He must then type the object name, terminated by a carriage return after pressing the key.

A map, obtained by pressing the Map key, presents a low resolution picture of the desk top. This image shows the position of all objects and navigational aids in the workspace. Map mode is ended by positioning the cursor at the spatial location desired and pushing the Goto key, causing the normal display to resume at the new location.

The system outputs any error messages on the Status line at the bottom of the display.

The largest components of the Automated Desk database ar the two arrays called image planes (i-planes) which contain the perspectives of the user's desk. All other AD structures support and index into these image planes.

4.2.4 Structure of the i-planes

Both the virtual terminal and icon image planes are large arrays containing alphanumeric characters. The text is organized into blocks of fixed size, called tiles. The tiles, arranged in a grid (Figure 4.3) are identified with two-word integers. The first word indicates the tile position as offset in the X-direction from the origin; the second word indicates the Y-offset. Thus, for example, if the tile closest to the origin is identified with the number (1,1), the tile located one position away in the X-direction

and two in the Y-direction would have the number (2,3). Notice that all directional offsets are positive.

Tile Grid

Figure 4.3

1,1	2,1	3,1	4,1	...	N-1,1	N,1
N+1,2	N+2,2	N+3,2	N+4,2	...	2N-1,2	2N,2
2N+1,3	2N+2,3	2N+3,3	2N+4,3	...	3N-1,3	3N,3

The lines represent tile boundaries and the numbers are the tile numbers.

Within each tile, positions are also referred to by their X- and Y-offsets from the origin of the tile. These offsets are contained in one byte. Therefore, three words are needed to express a position on the i-plane: two for the tile number and one for the position within the tile.

Each tile has a header which consists of:

- The tile number
- The modified flags
- The ready flag

The tile number identifies the tile. The modified flags are used by the navigator to determine if a tile must be written out to core or the disk when it is moved. The ready flag indicates that the entire file has been moved into core and is ready to use.

A tile is defined as containing enough data to cover one quarter of the user's screen. Thus, a tile consists of a 40 X 12 byte array (480 bytes), plus a 6-byte header.

Jumping between i-planes

An object on the virtual terminal plane is defined to be 80 X 24 character positions, the size of a full screen. The size of the standard icon is set to 20 X 6, a quarter in each dimension of a virtual terminal. However, the system can arrange icons in four sizes:

Standard	20 X 6
Tall	20 X 12
Wide	40 X 6
Large	40 X 12

The large icon causes four times the area in the virtual terminal plane to be reserved. The virtual terminal is centered in this area, and surrounded by blank space.

The space represented by one tile on the icon plane requires sixteen tiles on the virtual terminal plane. When translating the coordinates from those on the icon plane to those on the virtual terminal plane, the tile number and the X- and Y-offset within the tile number are all multiplied by four. The new tile number points to the upper left corner of a 16-tile rectangle in the virtual terminal plane. Unless the old center point was in the upper left sixteenth of the icon tile, the tile number must be incremented. The new X and Y components are divided by the tile dimensions and the quotients are added to the X and Y components of the tile number. The remainders are the new X and Y components.

4.2.4.1 The Kernel

The Kernel contains the utilities which support user-level functions. These utilities perform all manipulation of the image planes. The sections which compose the Kernel are:

- The Object Manager
- The Navigator
- The Feeder

4.2.4.2 The Object Manager

The Object Manager keeps track of available space in each of the i-planes. It accesses a list (called the object list) which contains the coordinates of each object in the i-plane. There are two object lists, one each for the icon and virtual terminal i-planes.

Tile maps aid the search in the object lists. A tile map is an array laid out similarly to the i-plane, with each tile represented by one word. A zero entry indicates the tile is completely empty.

The object list is sorted according to the left X-coordinate of each object. When the user adds an object, all entries whose left X-coordinate might cause the object to cover part of the area specified, must normally be searched to check the other coordinates. The tile pointer table was created to avoid this search. It contains pointers into the object list with all entries for a tile linked together. The entry in the tile map for a non-empty (occupied) tile is a pointer into this associated table.

As mentioned, each tile covers a quarter of a screen. Therefore, the Zentec needs only four tiles to produce the initial display. The terminal's display list is sent to point into this data which then shows on the screen.

As the user moves his viewing field, the screen borders no longer coincide with the tile borders (Figure 4.4). The system can draw from up to nine tiles to display the various portions of data on the screen at any one time.

Viewing Field   Figure 4.4

```
 _____________________________
|       |          |          |
|    ...|..........|...       |      Tile borders
 -----------------------------
|   .   |          |   .      |      Screen borders
|   .   |          |   .      |
 -----------------------------
|    ...|..........|...       |
|       |          |          |
 -----------------------------
```

4.2.4.3 The Navigator

The two i-planes are stored on a moving-head disk. The user views a small portion of one of the planes on the screen. The navigator is responsible for knowing where the user's viewing field is, and moving the surrounding data from disk to core to the terminal memory.

At system startup, the middle of the icon i-plane shows on the screen. Two transfers must occur before data will appear:

- From disk to core
- From core to the display

The navigator sends commands to a feeder to move specific data and the feeder performs the actual data transfer.
The icon core buffer holds an array of 10 X 12 tiles. This is a 57.5K byte buffer, or four times the data that can be stored in the display buffer. The navigator instructs the feeder to read the center of the icon i-plane (tiles (8,7) through 18,19) into core.

The center of the icon core buffer is then sent to the terminal. The terminal memory is 16K bytes, most of which is used to contain the data appearing on the screen. Some of this memory, however, is reserved for use by the program which does the scrolling.

Part of this reserved space is for the terminal display list. It has 24 entries, and each entry points to the start of a row that will appear on the screen. Thus, motion is accomplished by moving the pointers rather than the data itself.

The navigator continues to transfer data from the PDP-11/70 memory to the terminal after the system displays the initial view of the desk. In fact, it can place an array of 5 X 5 tiles in the terminal's data buffer, which allows a margin of another tile on all four sides, in anticipation of motion.

If the user moves in only one direction for very long, he will reach the edge of the display buffer. He cannot continue until the navigator sends new data to the Zentec.

Four thresholds (Figure 4.5) have been created so the user can move more smoothly across the desktop. If the screen data touches one of these thresholds, the navigator sends more information.

When the display data buffer is filled, space must be reused. The data on the side opposite to the direction of motion is least likely to be needed soon and is written over. (Modifications to the image plane occur only during object manipulation and output by active processes. Because the PDP-11/70 controls all of these, it updates the database at that time.)

Data stored on one side of the display data buffer can be moved to the opposite side merely by adjusting the left and right boundaries and using a circular buffer (Figure 4.6).

Screen Border Touching Navigator Thresholds   Figure 4.5

```
 ____________________________________
|      |                      |      |
|      |                      |      |
|      |                      |      |
------------------------------------ -- Top Y
|      |                      |      |
|      |                      |      |
|      |                      |      |
|      |                      |      |
|      |                      |      |
|      |                      |      |
|      |                      |      |
|      |                      |      |
------------------------------------ -- Bottom Y
|      |                      |      |
|      |                      |      |
|      |                      |      |
------------------------------------
       |                      |
       |                      |

       Left X                 Right X
```

There is one special case with the circular buffer which occurs when the seam of the buffer is in a row that is on the screen. The terminal refresh requires that the data in a row be contiguous. This is solved by adding an 80 character buffer at the end of the display data buffer.

Thresholds are also set up in the PDP-11/70 buffers. If the display data touches one of these thresholds, any tiles on the opposite side which have the modified flag set are written to disk and a new column is read in.

Movement Across Circular Display Data Buffer — Figure 4.6

This data:

11,10	12,10	13,10	14,10	15,10
11,11	12,11	13,11	14,11	15,11
11,12	12,12	13,12	14,12	15,12
11,13	12,13	13,13	14,13	15,13
11,14	12,14	13,14	14,14	15,14
11,15	12,15	13,15	14,15	15,15

becomes

12,10	13,10	14,10	15,10	11,11
12,12	13,11	14,11	15,11	11,12
12,12	13,12	14,12	15,12	11,13
12,13	13,13	14,13	15,13	11,14
12,14	13,14	14,14	15,14	11,15
12,15	13,15	14,15	15,15	11,10

4.2.4.4 The Feeder

The feeder performs the actual data transfer between the disk, core and the terminal. Most of the feeder resides in the PDP-11/70, but part of it is in the terminal ROM. The inputs to the feeder are:

- The upper left and lower right corners of the data to be moved.
- The i-plane.
- Where the data is coming from and where it is going to (disk, core, screen).
- An address for the upper left corner of the destination (for disk to core and core to terminal).

Moving from Disk to Core

The feeder obtains the tile numbers of the upper left and lower right corners of the rectangle to be moved and the address of the upper left corner of the rectangle presently in the core buffer.

The feeder determines the X and Y lengths of the rectangle to be moved by subtracting the coordinates at the upper left from those at the lower right. It calculates the address on the disk of the first tile in each row by the tile number. All tiles in a row are stored contiguously.

To find the disk address of a tile, the feeder determines that the byte offset from the start of the i-plane disk file is equal to the number of tiles in the X-direction of the i-plane, times the Y-component of the tile number, plus the X-component of the tile number, and multiplies everything times the number of bytes in a tile.

Rectangles to be moved from the disk are always in units of tiles. This convention has been adopted for simplicity.

The first tile starts at the given address in core. The feeder calculates succeeding addresses using the size of the rectangle being moved and the dimensions of the core buffer.

The modified and ready flags are always zero on the disk. The ready flag in a tile is set once the entire tile has been placed in core.

Moving from Core to Disk

Although a rectangle of data has been specified, only those tiles which have been modified are moved.

The ready flags in all the tiles in the rectangle are turned off so that none will be accessed during the transfer. Then the modified flag in each file is checked. For those that are to be moved, the modified flag is zeroed so that it will be initialized the next time the tile is placed in core; then the disk address is calculated and the actual transfer is done.

Moving from Core to the Terminal

The terminal-resident feeder which receives data is a program in the terminal ROM which, given the starting address and X- and Y-dimensions of the rectangle where the data is to be stored, reads the data sent by the PDP-11/70.

Processes

Each virtual terminal has a UNIX process associated with it. Two examples are NED, for document creation and review, and MSG, the electronic mail system. The user activates a process by centering on the virtual terminal and pressing the Activate key. The process remains active until one of the following occurs:

- The user centers of the virtual terminal and presses the De-activate key.
- The user sends a command to the process to end it.
- The process terminates itself.

A process can be moved offscreen and still remain active. However, most processes behave differently when offscreen. The user may have more than one active process at a time.

When he creates a virtual terminal, the user enters the names of two UNIX shell files or commands that are to be associated with the process. Shell files tailor the process to the specific virtual terminal. These files are:

- Startup script
- Shutdown script

When the Activate key is pressed the Startup shell file is invoked. An entry is also placed in the active process table. The table is used by the process monitor to determine when an active task has moved onscreen or offscreen.

Scripts for processes vary, but a Startup script is likely to initiate the process, giving it input and output files. Any input to the process is placed in the input file by one of the associated scripts (whichever one is currently active). A script also takes any data from the output file and sends it to either the display or an offscreen file. Any screen formatting (such as outputting the first page of a document) or initialization (such as polling a mailbox for new mail) is done by the Startup file. The Startup file initiates the execution of the onscreen file and then exits.

The Shutdown file is executed when a process terminates or the user de-activates it. It kills the UNIX job and removes the entry from the active process table.

The Process Manager

The process manager uses an active process list and two indices into it. An entry in the active process list consists of:

- A pointer to the virtual terminal's script names and offscreen options.
- The UNIX process ID of the process.
- The UNIX process ID of the current script.
- Left X of the virtual terminal.
- Top Y of the virtual terminal.
- Right X of the virtual terminal.
- Bottom Y of the virtual terminal.

The UNIX process-number of the process is used by the Shutdown script during de-activation.

The process number of the current script is used by the process manager to send the script a signal to start the other script and to exit.

The coordinates of the virtual terminal are needed to determine when the terminal starts to go offscreen and when it is fully onscreen.

The two lists which point into the active process table have the objects sorted, one by the X-coordinates and the other by the Y-coordinates. Since each virtual terminal has two of each, a virtual terminal is entered twice in both of the lists.

An entry in the X- (or Y-) coordinate list contains the X (or Y) coordinate and a pointer into the active process table.

4.3 Objects

Everything in the database is created and maintained by the user or by a process that he initiates. This section explains how the objects are created, updated, moved and deleted.

4.3.1 Creating Objects

The user has two function keys available for adding objects to the disk top: Create Object for a virtual terminal and associated icon, and Create Aid for a navigational aid.

Before any object can be entered, the object manager must determine if there is space for it. Only the icon plane need be checked if a new virtual terminal and icon are to be entered. If a standard size icon can be placed on the icon i-plane, then there is also room for the virtual terminal, since the two i-planes always have corresponding data. Further tests are made on the icon i-plane to determine if any larger size icons could be placed in the spot indicated by the cursor.

A navigational aid is placed on whichever i-plane the user is viewing when he requests the creation. Aids are always a standard size, so the object manager can easily determine if it can be entered.

If there is room for the object, it will be placed both in the i-plane and the indices into it. There are two indices for each i-plane -- the object list and the tile map.

The object list always contains the coordinates of the object in the i-plane and its name, if one has been assigned. An object list entry for a virtual terminal also contains data relevant to its associated process.

The tile map is used by the object manager to determine which areas of the i-plane are in use. An entry must be made in the tile map pointer table for each tile that the object is at least partially in. If the object created is the first one in the tile, the tile map pointer must also be added.

While an object is being created, the system cleans the user's screen. The system prompts him for the type of object. If the user is adding a virtual terminal and icon, the system prompts for the virtual terminal first. Once he has entered all the necessary parameters, the user returns to the section of the desk top he was viewing at the same perspective, with the new object visible.

4.3.1.1 Creating a Virtual Terminal

A virtual terminal is merely a screen-sized rectangle with a border until its process begins to send data to it. To create a virtual terminal, the user must choose the border character and object name (to be used later for rapid transit). Most importantly, he specifies the name of the associated UNIX process and the name of the scripts during these states:

- Startup
- Shutdown

These scripts are UNIX shell files which perform any initialization, check for special conditions and direct the input and output of the process.

All the data is stored in the virtual terminal object list. An entry contains:

- Virtual Terminal Name
- Left X
- Top Y

- Right X
- Bottom Y
- Process Name
- Startup Script Name
- Shutdown Script Name
- Spool File Name

4.3.1.2 Creating an Icon

An icon is a geometric shape with a distinctive border character around the text. The text serves as a key to the contents of the associated virtual terminal.

The user can determine the appearance of the icon, or he can leave it to the system. The system always uses a standard size rectangle with asterisks as the border character. If he wishes, the user can select his icon representation by having first the various shapes and then the various sizes of icons displayed on the screen. He picks one of each by centering on his preference.

The area reserved for an icon of any shape is that for a rectangle that would surround it. If the object manager determines that the large-sized rectangle could fit in the designated area, the user is shown all the icon sizes. However, if only the smaller icons will fit, the user is shown all of the shapes, but only in the valid sizes.

The user enters his choice of border character and the optional object name. The system displays the icon on the screen and the user can enter the inside text.

The icon is stored in the i-plane and entered in the tile map and object list. An entry for an icon in the object list consists of:

- Icon Flags
- Icon Name
- Left X
- Top Y
- Right X
- Bottom Y

4.3.1.3 Creating a Navigational Aid

A navigational aid has a fixed size and border. The only input from the user is the inside text. The shape is displayed on the screen and the user enters the marker's name. As with any other object the system enters the navigational aid in the i-plane, the object list and the tile map. A navigational aid object list entry consists of:

- Navigational aid flags
- Name
- Left X
- Top Y
- Right X
- Bottom Y

4.3.2 Updating Objects

Created objects can be changed at any time. The user centers on the object to be altered and presses the Update key. The system clears the screen and gives the prompts appropriate to the object type.

For example, when updating the characteristics of a virtual terminal, the user moves to the correct line by moving the directional keys and then types his new selection. The system knows which field he is changing by the cursor position. When the user exits, the system places the updated entry in the object list (the database for the border character).

If the user centers on an icon when he hits an Update key, the system prompts for icon changes. If the user wants to change the shape or size, the system clears the screen and displays the choices. The user selects one by centering on it, then he is returned to the icon update screen. The system updates the database when the user exits.

The only alteration a user can make to a navigational aid is to the label. In that case, the system clears the screen and displays the navigational aid currently stored. The user enters the new test and presses the Update key to exit.

4.3.3 Moving Objects

A user may want to move an object in the i-plane. Whether he moves an icon or a virtual terminal, the system moves the associated object on the other i-plane as well.

To move an object, the user places the cursor in the object he wishes to move and presses the <u>Pick</u> key. This places a pointer into the object list in the pick buffer. The user can then move around the i-plane, doing other functions if he wishes. When he locates the cursor where he wants the object, he hits the <u>Put</u> key. The system now calls the object manager. If the object is an icon or a virtual terminal, the object manager checks to see if there is room for the associated virtual terminal or icon. If there is no room, the object manager prints an error message on the control line. If there is room it moves the object in the database and updates the object list and the tile map.

4.3.4 Deleting Objects

If a user wishes to remove an object, he places the cursor in it and presses the Delete key. Deleting an icon or virtual terminal always erases the corresponding virtual terminal or icon.

The system removes the object from the i-plane, object list, and tile map pointer table. It adjusts the forward and backward pointers of any other objects in the tile or, if there are not other objects, resets the tile map word to zero.

4.4 Motion

Since the user cannot see his whole desk top on his screen, he will want to travel through it. He has the following methods:

- The directional keys on the terminal
- The navigational aid and Goto key
- The joystick

The directional keys on the terminal allow the user to move about within an i-plane. He can move in one of three directions: horizontally, vertically, or diagonally.

The system moves one character-position at a time. Initially, it moves one position; then, as the user holds the key down, it moves continuously until he releases the key.

The joystick can be used to produce the same type of motion, but gives the user more flexibility. The joystick interface sends the terminal an X, Y and speed (Z-plane) position each time the joystick position changes. The system processes joystick input through an interrupt handler, just as it processes keyboard input.

The (X, Y, speed) entry from either type of input is passed to the motion module that resides in the terminal ROM. This triple entry adjusts the terminal list pointers, which determine what appears on the screen.

The terminal list structure consists of 24 pointers, each of which points to the start of a row which appears on the screen. By altering the pointers, the terminal changes screen contents. For instance, if the X-input were one, the terminal would increment each list pointer by one. This would cause the left most column of the screen to scroll off a new column to appear on the right. If all the data is moved up one row, the second pointer replaces the first, etc. The terminal calculates a new 24th pointer (Figure 4.7) by adding length in the X-direction of the display data rectangle to the old 24th line (checking for wrap-around). The speed component determines the increment in the X- and Y-directions.

Display Screen   Figure 4.7

```
 ___________________
|                   |       Display data
|    --------       |
|    |      |       |       Screen data
|    --------       |
|    x              |       New 24th line pointer
|                   |
 -------------------
```

When the user centers an object, the system calculates how much adjusting must be done. It sends the X- and Y-corrections to the motion module in the same manner as keyboard or joystick input. The speed is set to one.

Although the user initiates map mode and <u>Goto</u> transit, the actual motion is caused by the PDP-11/70 sending new data to the terminal, and the center coordinates to the motion module. The center coordinates are used to calculate the borders of the screen. The module sets the first list pointer to the upper left corner of the screen data. Each

succeeding pointer has the length in the x-direction of the display data rectangle added to it, checking for wrap-around. This process also takes place during startup when the computer sends the first data to the terminal.

4.5 The Map

The map is a navigational aid which allows the user to establish his current location. The map presents a low resolution picture of the desk top.

The background of the map is always kept in core. When the user presses the map key, the navigator sends the map to the display buffer. The system calculates the position of X and displays that area of the map. The map remains on the screen until the user hits the Map key again. The navigator must always be testing for an active Map and send no data to the lower left quarter of the screen. The motion module must also notice if the X is to be moved, and, if so, move it.

4.6 Summary of Structures

The objects placed on the desk surface are stored in two disks resident as image planes. They contain text, which the system displays on the user's screen as he moves through the database.

The remaining structures have been created to manage the placement of objects in the images plane and their associated processes.

4.7 The Object Manager

When the user adds a new object, the system uses the following structures:

The tile map
The associated table
The object list

The tile map is laid out in a rectangle similar to the image plane, but with only one word for each tile. This word is zero if the tile is empty. Otherwise the word points in to the associated table. An entry in the associated table contains a pointer to yet another index called the object list; it also contains forward and backward pointers which link all entries for a single table. The object list points directly into the image plane. An entry contains the coordinates and the name, if one has been specified, of the object.

Since these are the only two objects on the desk surface, only two words in the tile map are non-zero (indicating occupied tiles). However, there are three entries in the associated table: two for the tile with the triangle and section of the rectangle, and one for the tile containing the remainder of the rectangle. Both of the entries for the rectangle point to the same object list entry.

Since the user may have many processes concurrently active, the system maintains an active process table. Not only does it add and remove processes as they are activated and de-activated, but their status changes as they move on-screen and off-screen. Thus, every time the user scrolls, the system must check the coordinates of all active processes to see if the motion is causing an active process to be fully on-screen or to begin to move off-screen. The system also maintains assorted coordinate indices into the active process

table, so the whole active process table need not be searched. One of these coordinate indices contains all the x-boundaries and the other y-boundaries. Since each object has two X- and Y- boundaries, each table is checked twice. If the user scrolls horizontally, the X-list must be checked; the Y-list is checked for vertical scrolling.

5. MCCRESSA

As part of the CTO/DDF program, CCA was involved in the technology transfer of MCCRESSA, the Marine Corps Combat Readiness Evaluation System Software Application, which evaluates the combat readiness of Marine units. Limited field tests of the prototype software were conducted by the original contractors. These limited field tests suggested improvements to be made to the software. More realistic tests, using actual Marine Corps personnel, identified and isolated many other ways to improve the operational version of the software. Areas for improvement included:

- User Interface
- Formatting of Data
- Completeness of Reporting
- Adequacy of Testing Breakdown

CCA conducted the field test of the actual performance of the prototype software. The effort included support in four areas: installation, training, support, and analysis. Each of these areas is discussed below.

5.1 Installation

CCA, in coordination with the hardware vendor, IBM, installed IBM 5110 systems at the following locations:

Camp Lejeune, North Carolina
Camp Pendleton, California
Marine Air Base, Hawaii
New Orleans, Louisiana
Twentynine Palms, California

5.2 Training

Extensive training exercises were carried out at these locations to obtain maximum utilization and exposure for the

software. Under separate contract with the United States Marine Corps, CCA conducted training sessions at CCA's facility in Washington, D.C.

CCA collected reports of problems with the prototype installation, evaluated them, and reported them to ARPA-CTO and Marine Corps Headquarters, along with appropriate recommendations. When possible, intermediate modifications of the prototype software were made. Changes and recommendations were documented so that the operational versions could also be modified.

Under a separate contract, an initial training activity took place at CCA's Washington facility. At least two representatives from each field facility were present. The training included discussion of fundamental concepts of the MCCRESSA software and the operations of the 5110 computer, and hands-on use of the system. CCA made required modifications to the user manuals for the prototype software for training purposes. In addition, field training covered system enhancements. This training was conducted concurrently with field visits made to evaluate the effectiveness of the system.

5.3 Analysis

CCA prepared criteria for evaluating the suitability of the product for operational use. Interviews, questionnaires, and observation of use in the field were used to determine recommendations for specific actions to be taken. An example of the type of criteria used to evaluate the system included the following:

- User Interface
- Command Entry
- System Response
- Error Handling

- Ease of Learning
- Flexibility

Each of these criteria in turn were broken down into more detail so that users could evaluate the system by answering very specific questions. The interviews and observations were used to determine more general, subjective evaluations.

As directed by the Marine Corps Commandant, field testing was measured using the technique specified by the MCCRESSA system. This technique hierarchically decomposes the tests and subsequent requirements associated with Marine activities. These requirements are defined in such a manner that they can be measured using a pass-fail measurement scheme. Data from the tests were aggregated by the MCCRESSA model to measure the readiness of tested Marine Corps units as compared at later dates with themselves and with other units. However, under current, limited testing, absolute measures of readiness have not yet been calibrated, though initial estimates of this calibration did result.

CCA believes that analysis of the results from the field test of the MCCRESSA software was an important first step in the technology transfer process since it was the first time that the software was exercised by actual operational personnel. Often, when users are presented with new tools for handling their jobs, they present resistance to the change. This field test and analysis of the prototype software encouraged the users by making it clear that their wants and needs were being considered in the design of the operational software. A key toward overcoming resistance was having people available to solve problems as they arose.

5.4 Summary

Limited field tests of MCCRESSA prototype software identified several deficiencies. Additional field tests with Marine Corps personnel pointed out other deficiencies and problems in such areas as the user interface, completeness of reporting, and others. These problems were corrected, thus improving and enhancing the operational version of MCCRESSA.

6. Dual DDF

As indicated in Section 1.3, the DDF not only reached a fully operational state earlier than originally planned; it also experienced heavier usage and supported a fuller demonstration schedule than was planned. The usage of the system for development and tests associated with technology transfer was itself very heavy. Furthermore, this usage and the demonstration schedules tended to conflict to some degree and disrupt one another.

To solve this problem, it became necessary to expand the DDF system configuration. The nature of the requirements for expansion and its results are the topic of the following subsections.

6.1 Problem

The rapid growth of DDF usage revealed several problems and requirements:

- Priority demonstrations required a dedicated system, and consequently interfered with development and transfer work.

- Certain programs, which were heavy consumers of processor time or memory, required a dedicated system.

- Certain basic R&D projects required an operating system other than UNIX, the DDF standard, and their own special-purpose peripheral gear which generated more demand for a dedicated system.

- Transfer progress was often slowed by non-prime time usage of the system.

- Extensive computing requirements were generated by Command System Cybernetics progıam map work.

All demonstrations caused some interference with regular DDF usage. The interference varied, depending on the products being demonstrated, from stopping other usage altogether to allowing only minimum usage monitored by the operator, to simply requesting that users exercise restraint in their consumption of resource. But because of the volume of actual demonstrations, the development of new demonstrations, and the continual increase of other usage, this interference became a significant problem.

6.2 Need

Progress in another dimension of DDF activity also increased computer resource demands: research product integration. As individual CTO research products were integrated into larger and larger packages for demonstration and transfer, there existed more and more occasions on which a demand for a dedicated processor would arise.

Sometimes in the software integration phase it became necessary to fall back on the original operating system from which a program was developed. This also resulted in additional dedicated system demand. The DARPA/CTO map work generated an especially heavy load of this type.

Finally, the preparation for transfer of CTO-developed software necessitated the blend of transferable FORTRAN IV programs to foreign operating environments. The DDF achieved success in this area by simulating those environments at the DDF. After intensive testing of the software under these conditions, the DDF could usually perform "one visit" transfers with few problems. This simulation and preparation effort also required dedicated usage of the computer.

Each of these requirements caused recurring interruption of the time-sharing service and inconvenience both to the time-sharing user community and DDF staff. Through planning, scheduling and appreciation of the immediate needs of CTO and its contractors, the DDF was able to shield users from some of the effects of this resource conflict. However, such shielding techniques could not satisfy the requirements in the face of steadily increasing demand.

6.3 Technical Approach

The requirements for an expanded DDF system configuration are described in the preceding paragraphs. The following paragraphs discuss the design of a configuration to meet those requirements.

6.4 Solution

As proposed, expansion of the DDF involved the addition of a second computer system, integrated into the existing configuration design so as to maximize resource sharing. First, this would permit the DDF to offer time-sharing service 24 hours a day, seven days a week. Time-sharing users would no longer be required to log-off the system for priority demonstrations or for the other reasons enumerated above. Second, dedicated system test and demonstration facilities would now be available as needed for:

A. Priority demonstrations.
B. Basic research.
C. Sophisticated product integration and transfer activities. In addition, the second system would provide an accessible facility for the maintenance of system software, the testing of specialized hardware devices, and the operation of programs with extraordinary resource demands.

Planners anticipated several beneficial side-effects with the addition of another mainframe.

- First, if ever the primary time-sharing system should crash due to hardware malfunction, the secondary research system could be brought up in its place. This would keep the affected users satisfied with their need for a constantly available system.

- Second, overhead and administrative work could be done on the second system. Tape dumps, system accounting, and system utilization runs could all be performed without interfering with user service.

- Third, priority one demonstrations could be run on a lightly loaded, or single-user, research system.

- With the addition of a second PDP-11/70, the DDF would have sufficient computer resources to fulfill the needs of the basic researcher, who might wish to change the equipment configuration or operating system for specific experiments, or need complete and uninterrupted utilization of the CPU for an extended period of time. The testing of new I/O drivers for transfer simulations, which would threaten the stability of time-sharing service, could be permitted on the research system.

- Finally, the DDF was asked to support the DARPA/CTO map work. This research project was being led by Perceptronics, Incorporated, and required large amounts of dedicated computer-time, hands-on access to the mainframe for two shift operations, and the addition of new equipment, such as the Evans and Sutherland Picture System 2, and the Evans and Sutherland Frame Buffer. This project was a major

addition to the DDF support of core activities and program development. Initial studies showed that there was insufficient third shift computer time available to accommodate DARPA/CTO map work without reducing time-sharing service.

6.5 Expanded Core Activities

The core activities were the basic services supplied by the DDF to CTO contractors. They include site management, PDP-11/70 time-sharing service, maintenance of the UNIX operating system, and system software, documentation, and other services. (Core activities are described at length in Section 3.)

The implementation of the dual DDF configuration resulted in an orderly expansion of the core activities of the DDF. CCA continued to use the physical site previously set up for the CM/DDF, incorporating the hardware and software configurations of the earlier experimental facility. Expansion of the basic services was designed to include the following upgrades:

- PDP-11/70 computer system
- One 10-ton EDPAC Air Conditioning Unit.
- Expansion of the computer room into the storage room including a new false floor and raising existing floor to nine inches.
- Expansion of power consumption and cables to support new computer equipment.
- Provision of a work area for the special map display equipment, including air conditioning and power for the GFE Evans and Sutherland Picture System 2 and Evans and Sutherland Frame Buffer.

6.6 Actual Implementation

After actual implementation of the Dual DDF began, a decision was made by ARPA/CTO to transfer the DDF to a new site. CCA therefore reformulated the Dual DDF plan to accommodate these changed circumstances. Specifically, CCA:

- Arranged for delivery and installation of the PDP-11/70 processor and peripherals at the new site, so that operations could begin there at an earlier date.
- Arranged for delivery and installation of the additional air conditioning at the new site.
- Transferred a disk drive to the new site.
- Coordinated the transition with the management of the new site as well as DDF users. The transition was accomplished smoothly, and operations began at the new site on schedule.

The CTO/DDF has operated with great success. Most of the problems encountered in the operations of the CTO/DDF resulted, paradoxically, from this success. The occasional poor response time, sub-optimal staff response time, and the exhaustion of available disk space were caused by the growth concomitant with success.

To future operators of facilities akin to the DDF, CCA offers the following wisdom:

- Plan for growth.
- Train your users. The typical user is driven by the need to complete a project, and will not invest in training unless strongly encouraged to do so; this results in inefficient use of the rapidly evolving capabilities of modern computers.
- Involve yourself in the design phase of users' projects. This improves the final product, and also allows better planning.

The implementation of the DDF concept accomplished under this contract was highly successful, and demonstrated a notable shortening of the "research to field use" transition for computer software.

7. References

[ANDRIOLE]
Andriole, S.V., Progress Report of an Integrated Crisis Warning System. Decision and Designs, Incorporated, Technical Report 76-19, December 1976.

[BROOKS]
Brooks, F. P. Jr., The Mythical Man-Month. Addison Wesley Publishing Company, 1975.

[CACI]
C.A.C.I., Incorporated-Federal, Executive Aid for Crisis Management - Sample Output. CACI Technical Report, November, 1977.

[LEAL ET AL]

[WITTMEYER]
Wittmeyer, J. R., Software Design for Interacting Crisis Early Warning Prototype System. Decision and Design, Incorporated, Technical Report 71-20, December 1976.